

**Nombre:** Víctor Alcántara.

**Matrícula:** 2023-1146.

**Carrera:** Desarrollo de software.

**Materia:** Programación III.

**Docente:** Kelyn Tejada Belliard.

**Tema:** Tarea 3 Parte teórica.

.

**Fecha:**  
21/11/2024

## Índice

<b>Introducción .....</b>	<b>3</b>
<b>1. ¿Qué es Git? .....</b>	<b>3</b>
<b>2. ¿Para qué funciona el comando git init? .....</b>	<b>3</b>
<b>3. ¿Qué es una rama? .....</b>	<b>3</b>
<b>4. ¿Cómo saber en cuál rama estoy? .....</b>	<b>4</b>
<b>5. ¿Quién creó Git? .....</b>	<b>4</b>
<b>6. ¿Cuáles son los comandos más esenciales de Git? .....</b>	<b>4</b>
<b>7. ¿Qué es Git Flow? .....</b>	<b>5</b>
<b>8. ¿Qué es Trunk Based Development? .....</b>	<b>5</b>
<b>Fuente bibliografica: .....</b>	<b>5</b>

# Introducción

Git es un sistema de control de versiones que permite gestionar de manera eficiente los cambios realizados en un proyecto, guardando un historial completo de todas las versiones anteriores para poder regresar a cualquier punto en el tiempo. Creado por Linus Torvalds en 2005, Git es ampliamente utilizado en el desarrollo de software, especialmente en proyectos grandes y colaborativos, ya que permite trabajar de manera independiente en diferentes partes del código mediante el uso de ramas. Con comandos básicos como `git init`, `git add`, `git commit` y `git push`, los desarrolladores pueden mantener un control preciso sobre sus contribuciones, mientras que enfoques como Git Flow y Trunk Based Development ayudan a organizar el flujo de trabajo en equipos.

## 1. ¿Qué es Git?

Git es un sistema de control de versiones, algo así como un "historial digital" de los cambios que haces en un proyecto. En vez de solo guardar la última versión de un archivo dígame por poner un ejemplo, Word o Excel, Git guarda todas las versiones previas/anteriores, para que puedas regresar a cualquier punto anterior en el tiempo, es digamos que divide el tiempo en varias fracciones o línea del tiempo jaja. Es digamos una herramienta que te permite llevar un registro de cada ajuste que realizas en tus proyectos de manera eficiente. Git es muy utilizado en programación porque ayuda a gestionar proyectos grandes, manteniendo un registro detallado de cada contribución. Es una forma de trabajar de manera colaborativa sin perder el control de las versiones.

## 2. ¿Para qué funciona el comando `git init`?

El comando `git init` es como abrir una nueva hoja en blanco para un proyecto. Lo que hace este comando es crear un repositorio vacío en la carpeta en la que lo ejecutes. Es el primer paso para comenzar a usar Git en tu proyecto. Este comando inicializa los archivos internos necesarios para que Git pueda empezar a rastrear y gestionar los cambios en tu proyecto. Básicamente, te da el poder de "activar" Git en una carpeta específica.

## 3. ¿Qué es una rama?

En Git, una **rama** (o "branch" en inglés) es como una "copia paralela" o digamos una rama diferente del árbol principal de tu proyecto en la que puedes hacer cambios sin afectar el trabajo principal. Puedes tener una rama para experimentar nuevas funcionalidades sin temor a interrumpir el trabajo que otros han hecho. Las ramas te permiten trabajar de manera independiente en diferentes partes del

proyecto y luego fusionarlas (merge) cuando todo esté listo esto es perfecto para digamos cuando trabajamos en grupo. Es como un sendero alternativo que puedes crear y explorar sin que afecte al camino principal.

#### 4. ¿Cómo saber en cuál rama estoy?











Para saber en qué rama estás trabajando, puedes usar el comando `git branch`. Este comando te mostrará digamos una lista de todas las ramas en tu repositorio, y la rama en la que te encuentras actualmente estará marcada con un asterisco (\*). También puedes usar `git status`, que te da información sobre tu estado actual en el repositorio, incluyendo la rama activa. Y si estas directamente en github es darle donde diga main y te aparecerán todas y ya.

#### 5. ¿Quién creó Git?

Git fue creado por **Linus Torvalds**, el mismo creador de Linux. Linus desarrolló Git en 2005 después de que el sistema de control de versiones anterior utilizado en el desarrollo de Linux fuera abandonado. Creo Git por la necesidad de un sistema rápido, eficiente y distribuido fue lo que lo motivó a crear Git, que hoy en día es uno de los sistemas de control de versiones más populares del mundo.

#### 6. ¿Cuáles son los comandos más esenciales de Git?

Para mi los mas esenciales son:

-  `git init`: Inicia un nuevo repositorio en tu carpeta.
-  `git clone [URL]`: Clona un repositorio remoto a tu máquina local.
-  `git add [archivo]`: Añade un archivo al área de "staging" para preparar el commit.
-  `git commit -m "mensaje"`: Guarda los cambios en el repositorio con un mensaje descriptivo.
-  `git status`: Muestra el estado actual de los archivos en tu repositorio (qué archivos están modificados, qué no se han comprometido, etc.).
-  `git pull`: Obtiene los cambios más recientes de un repositorio remoto.
-  `git push`: Envía tus cambios locales a un repositorio remoto.
-  `git branch`: Muestra las ramas existentes y la rama activa.
-  `git merge`: Fusiona una rama en la rama activa.
-  `git log`: Muestra el historial de commits.






## 7. ¿Qué es Git Flow?

**Git Flow** es un modelo de ramificación de Git que define una estructura de trabajo específica para desarrollar software de manera organizada. Básicamente, establece un conjunto de reglas sobre cómo gestionar ramas y flujos de trabajo en proyectos más complejos. El modelo de Git Flow propone tener ramas dedicadas para nuevas funcionalidades (feature), corrección de errores (hotfix), y un desarrollo estable (master o main), todo gestionado bajo un esquema que facilita la colaboración en equipos grandes. Es como un mapa para navegar a través del proceso de desarrollo.

## 8. ¿Qué es Trunk Based Development?

Trunk Based Development (TBD) es un enfoque de desarrollo de software en el que todos los miembros del equipo trabajan en una única rama principal, conocida como "tronco" (el tronco). En lugar de dedicar mucho tiempo a trabajar en ramas separadas, los desarrolladores hacen cambios rápidos y frecuentes en la rama principal, manteniendo el código siempre integrado y listo para producción. Este método fomenta la entrega constante, con una rápida integración de cambios para minimizar posibles conflictos de código. Consiste en mantener todos los caminos en un único sendero, en el que las personas avanzan en conjunto, adaptándose continuamente a los cambios.

## Fuente bibliografica:

-  <https://git-scm.com/doc>
-  <https://git-scm.com/docs/git-init>
-  <https://git-scm.com/docs/git-branch>
-  <https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git>
-  <https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository>